Universitaria de la Informática
ISBN: 978-99920-70-10-9

GitHub como herramienta docente

Francisco J Lopez-Pellicer, Rubén Béjar, Miguel A. Latre, Javier Nogueras-Iso, Fco Javier Zarazaga-Soria
Depto. de Informática e Ing. de Sistemas
Universidad de Zaragoza
{fjlopez,rbejar,latre,jnog,javy}@unizar.es

Resumen

El sistema de control distribuido de versiones Git se ha convertido en una herramienta esencial para manejar proyectos de software. Uno de los motivos de la creciente popularidad de Git es el éxito de GitHub, una plataforma Web de desarrollo colaborativo basada en Git. GitHub ofrece toda la funcionalidad de Git e integra diversas herramientas de control de acceso, colaboración, trazabilidad, gestión de tareas y control de proyectos. Recientemente, educadores dentro y fuera del mundo académico relacionado con la Informática han comenzado a usar GitHub en sus cursos. Esta contribución presenta una experiencia docente desarrollada en una asignatura relacionada con la ingeniería de software en la que GitHub se ha utilizado como la herramienta docente básica para el desarrollo de la parte práctica. Esta contribución se centra en motivar esta experiencia, explicar su implementación, evaluar los beneficios y riesgos potenciales que ha conllevado, e identificar nuevos retos.

Abstract

The distributed control version system Git has become an essential tool to manage software projects. One reason for the growing popularity of Git is the success of GitHub, a Web platform for collaborative development based on Git. GitHub offers all the functionality of Git and integrates various tools for access control, collaboration, tracking, task management and project management. Recently, educators inside and outside of the field of computer science have begun using GitHub in their courses. This contribution presents a teaching experience developed in a course related to software engineering where GitHub has been used as the basic platform for managing the assignments. This contribution focuses on motivating this experience, explaining its implementation, discussing the benefits and potential risks involved, and identify new challenges.

Palabras clave

Git, GitHub, plataforma colaborativa, transparencia, trazabilidad

1. Motivación

Los sistemas de control de versiones (SCV) son una herramienta esencial para manejar proyectos de software. Desde hace algunos años los SCV se han introducido en la enseñanza como herramienta docente. De hecho, cada vez es más fácil encontrase con trabajos que analizan su uso en ese contexto [2, 3, 8, 10, 11, 12, 13, 16].

Los SCV son herramientas de uso habitual en el desarrollo profesional de software desde hace décadas. Proporcionan una serie de funcionalidades claves para el desarrollo de proyectos como es el control de cambios en el código, la reversibilidad de dichos cambios, y la posibilidad de colaborar en el desarrollo del código. Además, los SCV permiten tener en paralelo varias versiones o ramas del proyecto. Las ramas se utilizan para desarrollar funcionalidades aisladas de los cambios en otras partes del proyecto que posteriormente pueden integrarse en la rama principal. Los SCV se pueden clasificar en dos grandes familias: SCV centralizados y SCV distribuidos. Los SCV centralizados como CVS[1] y Subversion[2] son sistemas cliente-servidor donde hay un repositorio canónico en el servidor que contiene toda la información de los cambios mientras que los clientes solo tienen copias de trabajo. En un sistema distribuido como Mercurial[3] y Git[4] no existe el concepto de repositorio canónico por lo que cada cliente ha de tener una copia completa del repositorio. Desde 2010, la tendencia es utilizar cada vez más SCV distribuidos, en particular Git [15].

Los SCV se están incorporando a la enseñanza por diferentes motivos: ayudar a desarrollar la competencia de trabajo en equipo [13], facilitar la retroalimentación de los alumnos [10], establecer escenarios de

[1] http://cvs.nongnu.org/
[2] http://subversion.apache.org/
[3] http://mercurial.selenic.com/
[4] http://git-scm.com/

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

desarrollo realistas [8] e, incluso, poder utilizarse como herramienta de monitorización [12]. Su uso no está exento de retos. Los alumnos pueden utilizarlo como un sistema de entrega más, sin aprovechar su funcionalidad [3], o usarlo de forma ineficiente [9]. Desde una perspectiva académica existe el riesgo de un aumento de las tareas de gestión docente [16] o que la curva de aprendizaje sea tan elevada que afecte al normal desarrollo del curso [14].

Esta contribución presenta la experiencia docente desarrollada en la asignatura *Ingeniería Web*[5] del itinerario de *Ingeniería de Software* del grado de *Ingeniería Informática* de la Universidad de Zaragoza durante el año 2014. En esta asignatura y para el desarrollo de la parte práctica, en lugar de utilizar directamente Git desde la línea de comandos se ha utilizado la plataforma y las herramientas de escritorio proporcionadas por GitHub[6] para afrontar los retos y riesgos anteriormente mencionados. GitHub es un servicio comercial de alojamiento en la Web de repositorios remotos Git. Está considerado como la plataforma de alojamiento de repositorios remotos más popular. A principios de 2014 el número de usuarios de GitHub se estimaba en más de 3,4 millones y el número de repositorios en 16,7 millones [20]. En abril de 2015, el número de usuarios es más de 9,4 millones y el número de repositorios alcanza los 22,4 millones [6]. Esta contribución se centra en motivar esta experiencia, explicar cómo se ha usado esta herramienta durante el curso, y evaluar beneficios, riesgos y retos.

2. Git y GitHub

2.1. Git

Git es un SCV distribuido diseñado para la gestión eficiente de flujos de trabajo distribuido no lineales. Git fue diseñado y desarrollado inicialmente por Linus Torvalds en 2005 para el desarrollo del kernel de Linux [1]. La licencia de Git es libre[7] y hay distribuciones oficiales[8] para los sistemas operativos Mac OS X, Windows, Linux y Solaris. La distribución de Git incluye herramientas de línea de comando y de escritorio. Además, hay disponibles herramientas proporcionadas por terceros que permiten una mayor integración con el escritorio[9] o con entornos de desarrollo[10].

La experiencia de Linus Torvalds en la gestión de la integración de las diferentes aportaciones en un proyecto distribuido de la magnitud del kernel de Linux

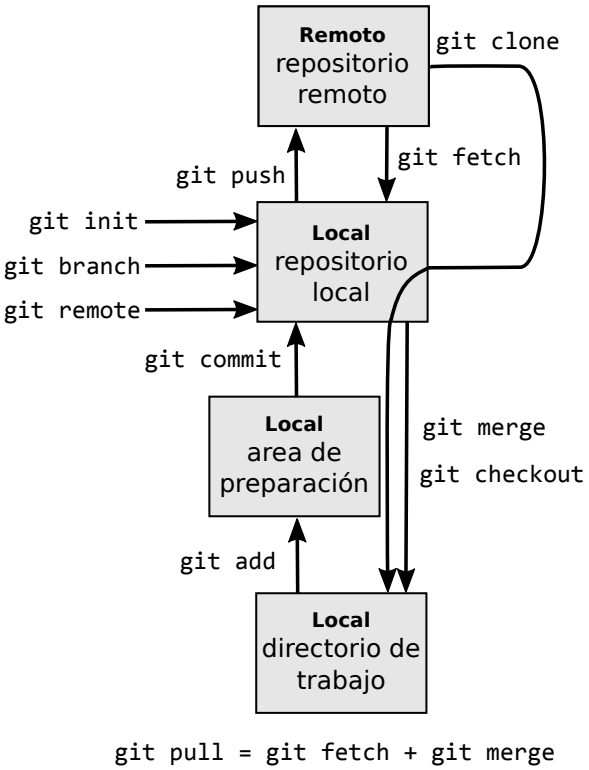


Figura 1: Principales comandos de Git

determinó las siguientes decisiones de implementación:

- *Versiones no incrementales*. Git almacena cada cambio como una instantánea de todos los archivos del proyecto. Para ser eficiente, si el archivo no ha sido modificado, sólo se almacena un enlace al archivo idéntico previamente almacenado. Los SCV anteriores a Git habitualmente almacenaban solo una versión base y las modificaciones hechas en cada cambio por archivo.
- *Autenticación criptográfica de la historia*. El identificador de un cambio se computa utilizando un algoritmo criptográfico que utiliza como entrada el cambio y la historia completa de cambios. Esto permite que cualquier cambio de la información durante la transmisión o en sistema de archivos sea detectado por Git.
- *Trabajo fuera de línea*. Por ser un sistema distribuido, cada repositorio de Git es un repositorio completo capaz de funcionar sin acceso a la red o al resto de los repositorios distribuidos gracias a que contiene una copia local de la historia completa del desarrollo del proyecto. Los cambios en la historia pueden copiarse de un repositorio a otro como nuevas ramas de desarrollo y se pueden copiar de la misma manera que una rama de desarrollo local.

[5] http://titulaciones.unizar.es/asignaturas/30246/index14.html
[6] https://github.com/
[7] https://gnu.org/licenses/gpl.html
[8] http://git-scm.com/downloads/
[9] http://git-scm.com/downloads/guis
[10] http://eclipse.org/egit/ por ejemplo

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

A continuación vamos a describir los principales comandos de Git (ver Figura 1). Para crear un *repositorio local* basta con ejecutar **`git init`** en un directorio. El repositorio creado tiene tres partes diferenciadas. El directorio es lo que se llama *directorio o espacio de trabajo*, y dentro de la carpeta *.git* que ha sido creada por el comando se encuentra el *área de preparación*, que actúa como una zona intermedia, y el *historial de cambios*. El repositorio local también se puede crear haciendo una copia de otro repositorio local o remoto mediante el comando **`git clone`**. Los ficheros del directorio de trabajo cuyos cambios se quieren controlar se registran con el comando **`git add`**. Para confirmar estos cambios se ejecuta **`git commit`**. Git permite el uso de ramas (*branches*). Por defecto se trabaja en la rama denominada *master*. El comando **`git branch`** permite crear y destruir ramas. Para cambiar de rama en el directorio de trabajo se utiliza el comando **`git checkout`**. Si se desean enviar los cambios a un repositorio remoto se utiliza el comando **`git push`** indicando la rama cuyos cambios se quieren enviar. Previamente los repositorios remotos se han configurado utilizando el comando **`git remote`**. Para actualizar el repositorio con los cambios de un repositorio local se utiliza el comando **`git pull`**. Este comando es esencialmente un encadenamiento de dos comandos: **`git fetch`** y **`git merge`**. El primero obtiene los cambios de una rama remota. El segundo fusiona si es posible estos cambios con una rama local. El comando **`git pull`** es un ejemplo de la orientación a caja de herramientas que caracteriza el diseño de Git. Git está implementado como un conjunto de programas y scripts de shell que son fácilmente encadenables para formar nuevos comandos. Además, Git cuenta con mecanismos para lanzar *scripts* de usuario cuando suceden ciertos eventos en el flujo de trabajo denominados puntos de enganche (*hooks*).

Según [19], si en 2011 más de la mitad de los programadores encuestados utilizaban como herramienta principal el SCV centralizado Subversion (51,3 %) y solo un 12,8 % utilizaba Git, hoy casi la mitad de los programadores usan Git o GitHub (41,9 %) seguido de un 30,7 % que siguen usando Subversion.

Este cambio ha afectado también al perfil profesional demandado por los empleadores. Por ejemplo, en el portal ITJobsWatch[11] especializado en analizar las ofertas laborales en el Reino Unido se observa que ser competente en Git es mucho más demandado que ser competente en Subversion (23,3 % versus 16,1 % en enero de 2015). No solo eso, la demanda de personal técnico competente en Git está superando los máximos históricos de Subversion (20,0 % en marzo de 2012).

[11] `http://www.itjobswatch.co.uk/`

2.2. GitHub

GitHub es un servicio comercial de alojamiento de repositorios Git remotos creado en el año 2008. GitHub proporciona una interfaz Web que permite al usuario registrado crear repositorios vacíos o por clonación de otro repositorio hospedado en GitHub (*fork* en la terminología de GitHub), enviar solicitudes de cambio entre repositorios hospedados (*pull request* en la terminología de GitHub), y gestionar dichas solicitudes. Además, los repositorios hospedados en GitHub pueden actuar como repositorios remotos de repositorios locales. Los repositorios que se crean en GitHub son por defecto de acceso público. Solo mediante una cuenta de pago o si cumplen ciertas condiciones es posible alojar repositorios privados. Cada repositorio de GitHub es propiedad de una cuenta de usuario o de organización. Las modificaciones a un repositorio alojado en GitHub solo se pueden realizar por usuarios que han iniciado sesión y que estén autorizados a modificar su contenido, o a aquellos autorizados que han configurado un repositorio local de Git para que con las credenciales proporcionadas por GitHub pueda conectarse a dicho repositorio.

Además del alojamiento, GitHub proporciona a cada repositorio una wiki, un gestor de tareas (*issues*), un completo sistema de gestión de comentarios, un cuadro de control con grafos sociales e, incluso, una página web propia. Finalmente, todo este contenido puede ser accedido y manipulado mediante un API Web[12].

El éxito de GitHub señalado en la motivación de este trabajo ha atraído la atención de los investigadores. Por ejemplo, [4] ha analizado las dinámicas de trabajo en los proyectos de código abierto en un entorno tan transparente detectando mejoras en el compromiso, calidad del trabajo y reconocimiento personal de las personas involucradas.

Si nos centramos en el campo educativo, hay que destacar que GitHub ha creado cuentas de usuario especiales para profesores, estudiantes y centros educativos[13]. El Cuadro 1 nos muestra ejemplos de cursos con repositorios públicos en GitHub. Se estima que en 2014 había más 1200 cursos y 70.000 estudiantes utilizando este tipo de cuentas [17]. Por ello no es de extrañar el creciente interés en analizar su uso como herramienta educativa tanto para impartir contenidos informáticos [7, 22] como no informáticos [18].

Hay que hacer notar que GitHub no es un proyecto de código abierto. Sin embargo, existen actualmente soluciones de software abierto como Gitorious[14] y GitLab[15] que permiten instalar en servidores propios un servicio de alojamiento de repositorios remotos con

[12] `https://developer.github.com/v3/`
[13] `https://education.github.com/`
[14] `https://gitorious.org/`
[15] `https://about.gitlab.com/`

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

Universidad	*Tópico*	*Uso*	*Desde*	*URL*
U Chicago (EE.UU.)	Varios cursos	Material de docencia	2012	`https://github.com/uchicago-cs`
U California Davis (EE.UU.)	Genómica	Material de docencia	2013	`https://github.com/RILAB`
HiOA (Noruega)	Varios cursos	Material de docencia	2013	`https://github.com/hioa-cs`
U Politécnica Madrid (España)	Linked Data	Prácticas	2013	`https://github.com/FacultadInformatica-LinkedData`
U Complutense de Madrid (España)	Procesadores de Lenguajes	Prácticas	2014	`https://github.com/plgucm`

Cuadro 1: Ejemplo de uso de repositorios públicos de GitHub en la docencia

herramientas similares a las que ofrece GitHub.

3. GitHub aplicado a la docencia

Ingeniería Web es una asignatura de 6 créditos ECTS del cuarto curso del grado de *Ingeniería Informática* de la Universidad de Zaragoza. La asignatura es parte del itinerario de *Ingeniería del Software*. A lo largo de la asignatura el alumno debe desarrollar en solitario o en grupo varios sistemas Web orientados a servicios, desarrollando así competencias relacionadas con el desarrollo Web. Ingeniería Web es una asignatura de evaluación continua en la que el trabajo práctico pondera un 80 % en la nota final.

A continuación describimos herramientas y funcionalidades de GitHub utilizadas en este curso distinguiendo entre las derivadas de Git y las exclusivas de GitHub. Las herramientas y funcionalidades se presentan agrupadas bajo tres criterios. El primero atiende a su uso por parte del alumno en su aprendizaje. El segundo se fija en su papel como soporte del curso. Finalmente, el tercero denominado facilitadores describe aspectos relacionados con tecnologías, licencias y precios que han facilitado el diseño curso.

3.1. Aprendizaje del alumno

Dentro de esta categoría incluimos las herramientas y funcionalidades relacionadas con actividades de comunicación, productividad y participación de los alumnos. Las derivadas de Git son:

- *Diario de actividades*. El histórico de cambios de un repositorio hospedado en GitHub es accesible vía Web, pudiendo accederse a todos los cambios que se han producido con indicación de quién lo ha cambiado, cuándo se ha cambiado, qué ha cambiado y el origen del cambio (otro repositorio o vía un editor Web proporcionado por GitHub). Toda la información presentada es navegable.
- *Trabajo fuera de línea*. Implícito por soportar Git pero restringido a los recursos gestionados en los repositorios por Git. No hay soporte fuera de línea a las herramientas de trabajo colaborativo de GitHub.
- *Trabajo en grupo*. El uso de Git y GitHub permite implementar diferentes flujos de trabajo en entornos educativos [21]. En esta asignatura se ha organizado según se describe en la Figura 2.

La relevancia del *trabajo en grupo* en esta asignatura hace que merezca la pena detallar su implementación. Primero hay que preparar los repositorios remotos compartidos de los grupos y los repositorios remotos y locales de los alumnos. GitHub actúa como la máquina remota donde se crean los repositorios remotos compartidos. A continuación, cada alumno clona en GitHub el repositorio remoto compartido de su grupo para crear su repositorio remoto. El siguiente paso es la creación por parte de cada alumno de un clon local de su repositorio remoto utilizando Git. Al repositorio local se le añade como remoto el repositorio remoto del grupo. El repositorio remoto del alumno se referencia localmente en Git como *origin* y el del grupo como *upstream*. Una vez que está preparada toda la infraestructura, el flujo de trabajo esperado de cada alumno sigue los siguientes pasos: integrar los cambios del repositorio del grupo en su repositorio local (**`git pull --rebase upstream`**), realizar los cambios pertinentes, enviar dichos cambios a su repositorio individual remoto (***`git push --force origin`***) y, finalmente, solicitar vía un *pull request* que los cambios de su repositorio remoto se integren en el repositorio remoto del grupo. Dentro del grupo hay un alumno con el rol gestión del repositorio remoto. Este alumno es el que decide si se integra el cambio o no en el repositorio del grupo.

La única herramienta de GitHub que puede ser considerada de aprendizaje no asociada con Git y de la cual se tiene constancia de su uso por parte de los alum-

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

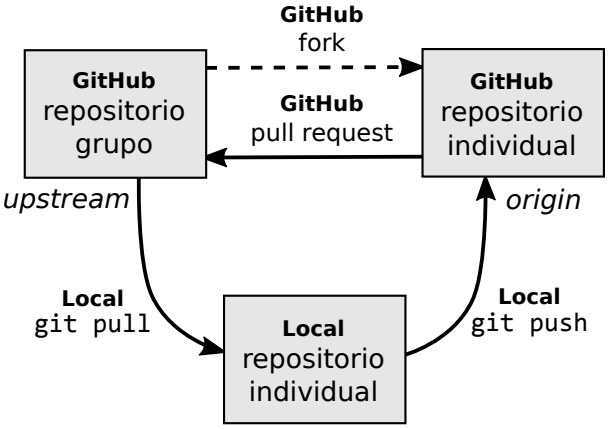


Figura 2: Flujo de trabajo en grupo utilizando GitHub/Git

nos son los *micro foros de discusión*. Los alumnos que han iniciado sesión en GitHub pueden añadir vía Web comentarios a cualquier cambio y a los *issues* asociados a un repositorio público. También permite al propietario del repositorio (por ejemplo, un profesor) moderar dichos comentarios. Esta herramienta se ha utilizado principalmente en la interacción profesor-alumno durante las entregas. Otras herramientas como el potente *motor de búsqueda facetado* que puede restringir la búsqueda a usuarios, repositorios, rutas, lenguajes de programación, asuntos, y fechas de creación y mezcla, puede haber sido utilizado por los alumnos pero no existe forma de cuantificar su uso.

3.2. Soporte del curso

Dentro de esta categoría incluimos las herramientas y funcionalidades relacionadas con actividades el soporte del curso, incluyendo aspectos administrativos, de desarrollo del contenido y de realización del curso. Las derivadas de Git son:

- *Integridad*. Tal como se ha comentado en una sección anterior, Git incorpora una autenticación criptográfica del historial de cambios de un repositorio. Esta característica implica que un alumno no puede modificar un cambio integrado en la historia del repositorio para añadir o borrar un fichero o alterar su contenido. En consecuencia, el histórico de cambios es una imagen fiel de la actividad del alumno o del grupo.
- *Compartir material de prácticas*. El material de prácticas del curso así como todo el código que necesitan los alumnos está almacenado en repositorios públicos. Se ha enfatizado a los alumnos que deben clonar el material de prácticas. De esta manera, ya sea por nuevo material disponible o para corregir errores detectados, pueden aprovechar Git para incorporar los cambios.

Las herramientas y funcionalidades exclusivas que ofrece GitHub son:

- *Transparencia.* [4] señala que la disponibilidad en GitHub de pistas visibles como el volumen de actividad, la actividad a lo largo del tiempo, la relevancia del flujo de cambios, y la información detallada son herramientas útiles para resolver problemas de coordinación y comunicación. Aplicada a la gestión del curso, la transparencia ha resuelto problemas de coordinación y comunicación entre los profesores y los alumnos, y entre los alumnos entre sí.
- *Publicación de contenido*. GitHub genera automáticamente un sumario en HTML del contenido del repositorio o de una carpeta si existe un fichero denominado *README* o *README.md* (.md es la extensión que corresponde al lenguaje de marcado ligero Markdown[16]).
- *Edición en línea de contenido*. GitHub permite editar contenido vía un editor Web. Según el tipo de extensión, por ejemplo *.md*, permite su pre visualización.
- *Seguimiento de los alumnos*. Es posible acceder a diferentes visiones de la actividad de los alumnos durante el curso en forma de diferentes tipos de informes (ver Figura 3): informes globales de actividad, seguimiento del trabajo en equipo, informes comparativos de actividad e informes individuales de actividad.
- *Plataforma de entrega con realimentación*. La combinación del uso de *pull requests* como herramienta de entrega junto con los micro foros de discusión permiten controlar la entrega y realimentar al alumno con el resultado. Este procedimiento es uno de los recomendados por GitHub cuando se utiliza como plataforma de entrega en la docencia[17].

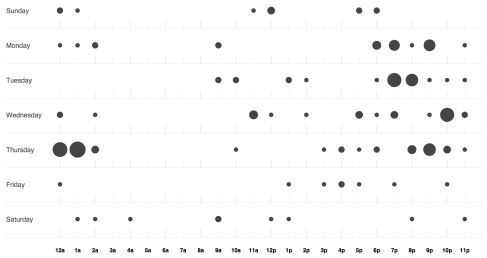
3.3. Facilitadores

Dentro de esta categoría incluimos aquellas características de GitHub relacionadas con tecnologías, licencias y precios que han facilitado el diseño del curso. Tres facilitadores están directamente derivados de Git:

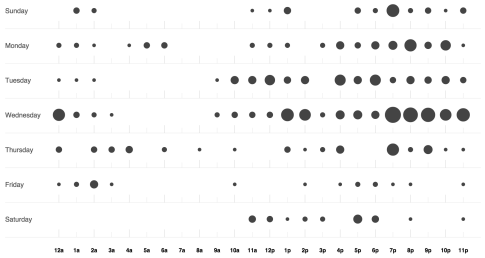
- *Multiplataforma*. Existen distribuciones oficiales para los sistemas operativos Mac OS X, Windows, Linux y Solaris. Los tres primeros cubren la totalidad de los sistemas operativos utilizados por los alumnos. La distribución de Git incluye herramientas de línea de comando y de escritorio.
- *Clientes pesados (de Git)*. Hay disponibles clientes proporcionados por terceros que permiten una

[16] `http://daringfireball.net/`
[17] `https://education.github.com/guide/assignments`

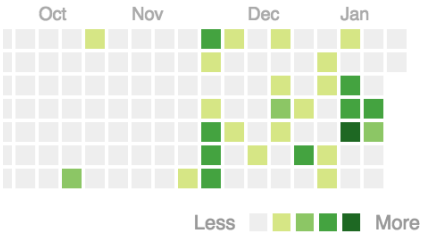
Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9



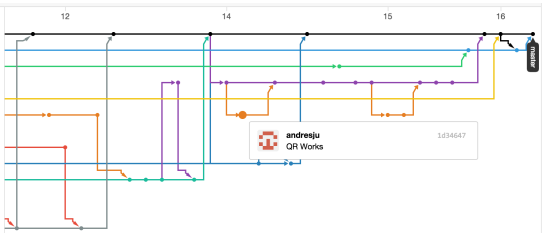
(a) *Punch card*: Comportamiento al notificar la entrega de las prácticas obligatorias



(b) *Punch card*: Comportamiento al integrar cambios durante del proyecto final



(c) *Public contributions*: Sumario gráfico con datos cuantitativos/cualitativos de actividad enlazados



(d) *Network*: Seguimiento e interacción dentro de los equipos

Figura 3: Informes proporcionados por la plataforma GitHub

mayor integración con el escritorio o con entornos de desarrollo. Esto simplifica el uso de Git por parte de los alumnos.

- *Código abierto*. Aun cuando la plataforma GitHub es comercial, la herramienta que se utiliza en el lado del cliente es Git, cuya licencia es libre y da acceso al código. Es decir, no hay un coste oculto por instalar Git en los equipos que van a usar los alumnos.

Otros tres facilitadores son exclusivos de GitHub:

- *Clientes pesados (de GitHub).* GitHub proporciona clientes pesados multiplataforma. Además, muchos entornos de desarrollo incluyen extensiones que además de trabajar con Git pueden interactuar con GitHub mediante su API. Los alumnos han trabajado habitualmente a través de estos clientes pesados.
- *Interfaz Web.* La interfaz Web de GitHub es de fácil uso y reduce el coste del aprendizaje de Git. Los alumnos no han manifestado incomodidad al usarlo.
- *Coste del servicio*. Se han utilizado cuentas públicas gratuitas. Aun cuando para estudiantes existe la posibilidad de solicitar cuentas especiales, estas no han sido necesarias.

4. Discusión

GitHub, tal como se ha utilizado en esta asignatura, presenta beneficios tanto para el aprendizaje de los alumnos como para la gestión de la asignatura. El alumno puede adquirir una serie de competencias profesionales transversales relacionadas con la capacidad para usar herramientas de la Ingeniería Informática (uso continuado de un SCV) y con el trabajo en grupo (coordinación de trabajo distribuido). Queda en segundo plano, ya que no es un objetivo principal de la asignatura, el desarrollo competencias relacionadas con la aplicación de los principios y metodologías de la Ingeniería de Software, en particular la gestión del proyecto utilizando las herramientas proporcionadas por GitHub. Un beneficio indirecto y a más largo plazo es el uso de una herramienta que servirá al alumno para comunicar a terceros sus habilidades y destrezas profesionales ya que la cuenta de GitHub puede convertirse en su portafolio profesional. Desde el punto de vista del soporte del curso, GitHub ha permitido distribuir el contenido de prácticas con los alumnos de una forma más cercana a los usos y costumbres asociados al tópico de la asignatura (Ingeniería Web). El flujo de trabajo y entrega así como los diferentes informes que proporciona la plataforma han permitido hacer un seguimiento de las actividades de los alumnos. Además, cuando se ha detectado que era necesario, las herramientas de comunicación disponibles se han utilizado para señalar y corregir problemas potenciales relacio-

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

nados con el desempeño de los alumnos.

No obstante, el uso de GitHub en la docencia no está exento de riesgos. Las costumbres adquiridas por los alumnos son difíciles de cambiar por lo que competencias anteriormente identificadas pueden no ser adquiridas. Por ejemplo, no todos los alumnos han utilizado GitHub como la herramienta de coordinación del trabajo distribuido. Algunos han seguido utilizando servicios como Dropbox para coordinar su trabajo y, una vez finalizado, han utilizado GitHub solo como plataforma de entrega. Otro riesgo asociado al uso de GitHub es la dependencia de una plataforma comercial. Este riesgo se mitiga en parte por la disponibilidad de plataformas de código abierto como Gitorious y GitLab que podrían ser instaladas en equipos docentes. Finalmente, un tercer riesgo que el lector puede percibir es la posibilidad de plagio al trabajar con repositorios públicos en tareas evaluables. En nuestro caso concreto no es un problema tal como se plantearon las prácticas. Los proyectos evaluables tenían características notablemente diferenciadas por lo que un mero plagio no era una estrategia viable por parte de los alumnos. En cualquier caso, se puede plantear la integración herramientas que detecten plagios en el código, como Moss[18], mediante la creación de scripts que se ejecutarían asociados a operaciones de Git vía *hooks*.

5. Conclusiones

La plataforma GitHub (u otra similar) aplicada a la docencia no debe ser considerada como un mero servicio de alojamiento de repositorios de Git en la Web con funcionalidades extra. En la experiencia docente descrita en este trabajo hemos comenzado a vislumbrar su potencial papel como herramienta de aprendizaje y de gestión de la enseñanza. Es necesario profundizar en el análisis de esta plataforma como herramienta docente para dimensionar adecuadamente su papel. Futuras investigaciones deben abordar aspectos cuantitativos y cualitativos relacionados, por ejemplo, con la satisfacción de alumnos y profesores, los resultados académicos, y la evolución de la carga docente.

En cualquier caso, la aparición de plataformas como GitHub plantea nuevos retos a la docencia como, por ejemplo, su integración con las plataformas educativas existentes, el desarrollo de herramientas de apoyo orientadas a la gestión educativa, la gestión de nuevos retos relacionados con la propiedad intelectual y la ética profesional, y el desarrollo de buenas prácticas para un correcto aprovechamiento de esta nueva generación de herramientas de docencia.

[18] `http://theory.stanford.edu/~aiken/moss/`

Agradecimientos

Este trabajo ha sido financiado a través del proyecto PIIDUZ_2014_127 correspondiente a las ayudas del Programa de Incentivación de la Innovación Docente en la Universidad de Zaragoza.

Referencias

[1] Scott Chacon y Ben Straub. *Pro Git*. Apress, 2nd edición, Noviembre 2014, ISBN 9781484200773.

[2] Curtis Clifton, Lisa C Kaczmarczyk y Michael Mrozek. *Subverting the fundamentals sequence*. ACM SIG on Computer Science Education Bulleting, 39(1):86, Marzo 2007.

[3] Michael Cochez, Ville Isomöttönen, Ville Tirronen y Jonne Itkonen. *How Do Computer Science Students Use Distributed Version Control Systems?* En Vadim Ermolayev, HeinrichC Mayr, Mykola Nikitchenko, Aleksander Spivakovsky y Grygoriy Zholtkevych (editores): *Metadata and Semantic Research*, páginas 210–228. Springer International Publishing, 2013, ISBN 978-3-319-03997-8.

[4] Laura Dabbish, Colleen Stuart, Jason Tsay y Jim Herbsleb. *Social coding in GitHub*. En *ACM 2012 conference on Computer Supported Cooperative Work*, páginas 1277–1286, New York, New York, USA, 2012. ACM Press, ISBN 9781450310864.

[5] Vadim Ermolayev, Heinrich C Mayr, Mykola Nikitchenko, Aleksander Spivakovsky y Grygoriy Zholtkevych (editores). *Information and Communication Technologies in Education, Research, and Industrial Applications*, volumen 469 de *Communications in Computer and Information Science*. Springer International Publishing, Cham, 2014, ISBN 978-3-319-13205-1.

[6] GitHub. *Press*, 2015. `https://github.com/about/press`.

[7] Terry Griffin y Shawn Seals. *GitHub in the classroom: not just for group projects*. Journal of Computing Sciences in Colleges, 28(4):74–74, Abril 2013.

[8] Ken T N Hartness. *Eclipse and CVS for group projects*. Journal of Computing Sciences in Colleges, 21(4):217–222, Abril 2006.

[9] Ville Isomöttönen y Michael Cochez. *Challenges and Confusions in Learning Version Control with Git*. En *Information and Communication Technologies in Education, Research, and Industrial Applications*, páginas 178–193. Springer International Publishing, Junio 2014, ISBN 978-3-319-13205-1.

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

[10] Oren Laadan, Jason Nieh y Nicolas Viennot. *Teaching operating systems using virtual appliances and distributed version control*. En *the 41st ACM technical symposium*, página 480, New York, New York, USA, 2010. ACM Press, ISBN 9781450300063.

[11] Joseph Lawrance, Seikyung Jung y Charles Wiseman. *Git on the cloud in the classroom*. En *44th ACM technical symposium*, página 639, New York, New York, USA, 2013. ACM Press, ISBN 9781450318686.

[12] Ying Liu. *CVS historical information to understand how students develop software*. Ïnternational Workshop on Mining Software Repositories (MSR 2004)"W17S Workshop - 26th International Conference on Software Engineering, 2004:32–36, Enero 2004.

[13] Andrew Meneely y Laurie Williams. *On preparing students for distributed software development with a synchronous, collaborative development platform*. En *40th ACM technical symposium on Computer science education*, página 529, New York, New York, USA, Marzo 2009. ACM Request Permissions, ISBN 9781605581835.

[14] Ivan Milentijevic, Vladimir Ciric y Oliver Vojinovic. *Version control in project-based learning*. Computers & Education, 50(4):1331–1338, Mayo 2008.

[15] Stephen O'Grady. *DVCS and Git Usage in 2013*, Diciembre 2013. `http://redmonk.com/sogrady/2013/12/19/dvcs-and-git-2013/`.

[16] Karen L Reid y Gregory V Wilson. *Learning by doing: introducing version control as a way to manage student assignments.* SIGCSE, páginas 272–276, 2005.

[17] Paul Sawers. *GitHub Wants Schools to Collaborate Around code*, Febrero 2014. `http://thenextweb.com/insider/2014/02/11/github-wants-schools-collaborate-code/`.

[18] Kris Shaffer. *Push, Pull, Fork: GitHub for Academics*. Hybrid Pedagogy, 2013. `http://www.hybridpedagogy.com/journal/push-pull-fork-github-for-academics/`.

[19] Ian Skerrett. *Eclipse Community Survey 2014 Results*, Junio 2014. `https://ianskerrett.wordpress.com/2014/06/23/eclipse-community-survey-2014-results/`.

[20] Marisa Whitaker. *GitHub co-founder Chris Wanstrath shares his story*, Abril 2014. `http://magazine.uc.edu/content/magazine/favorites/web-only/wanstrath.html`.

[21] Zhiguang Xu. *Using Git to Manage Capstone Software Projects* . En *7th International Multi-Conference on Computing in the Global Information Technology*, 2012.

[22] Alexey Zagalsky, Joseph Feliciano, Margaret Anne Storey, Yiyun Zhao y Weiliang Wang. *The Emergence of GitHub as a Collaborative Platform for Education*. ACM, New York, New York, USA, Febrero 2015, ISBN 978-1-4503-2922-4.